Inventor: Tomoyoshi Sato
Title: "Data Processing System"
U.S. Serial No.: Not Yet Assigned
Atty. Docket No.. 29898/40913
Marshall, Gerstein & Borun LLP 312-474-6300
Sheet 1 of 12

Fig. 1

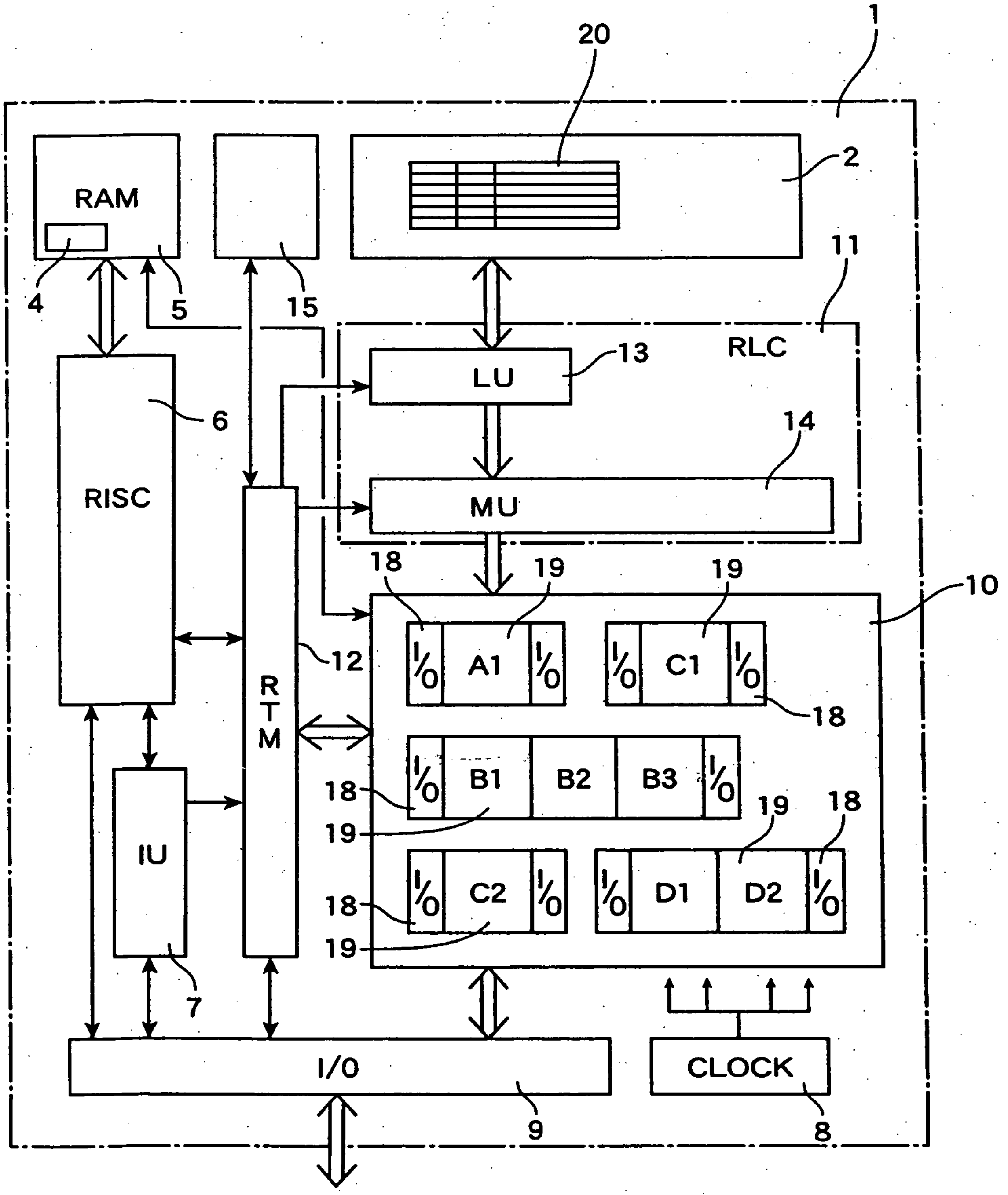


Fig. 2

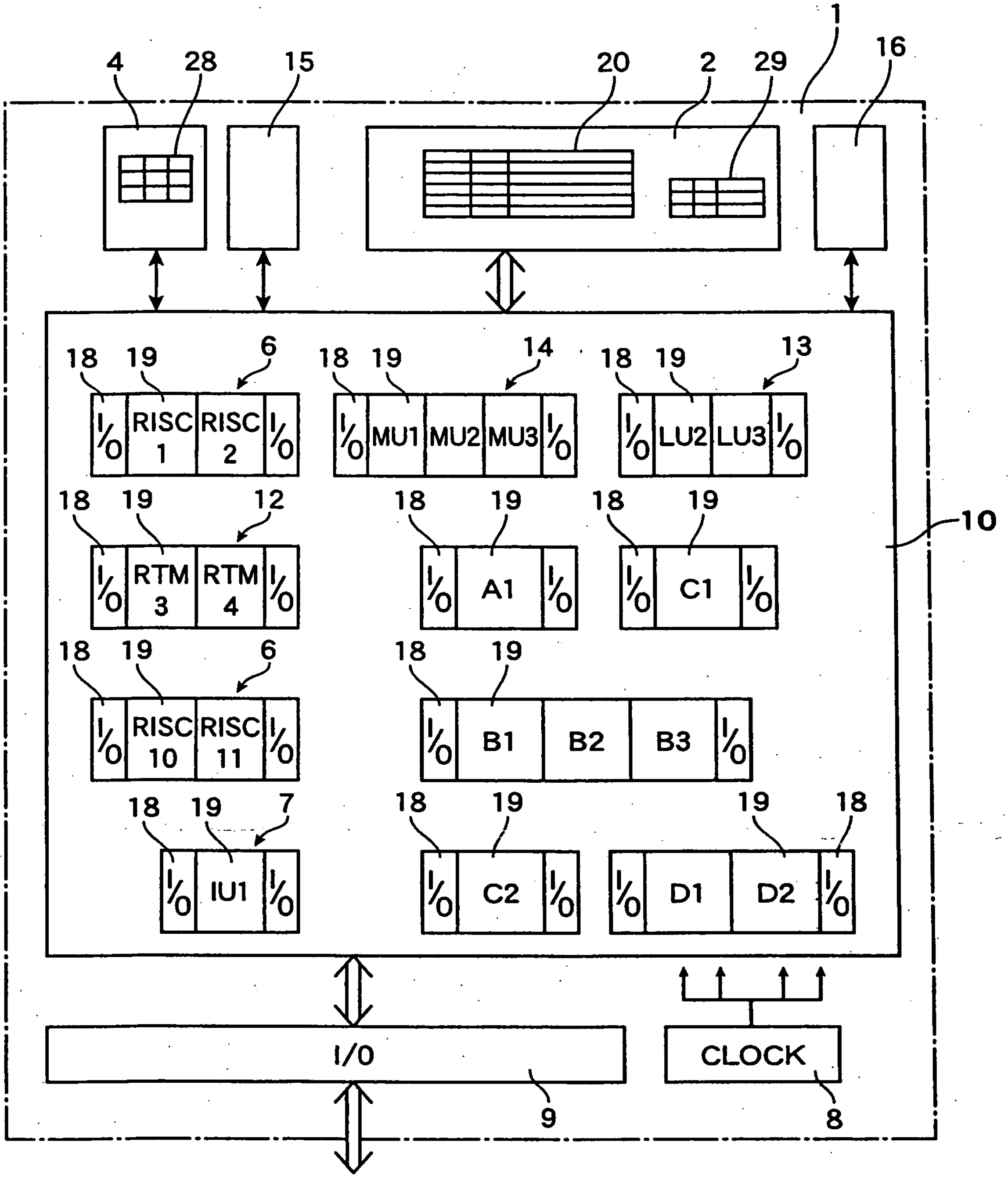

1
4
28
15
20
2
29
16
10
18
19
6
RISC 1
RISC 2
I/O
14
MU1
MU2
MU3
13
LU2
LU3
12
RTM 3
RTM 4
A1
C1
RISC 10
RISC 11
B1
B2
B3
7
IU1
C2
D1
D2
I/O
CLOCK
9
8

Fig. 3

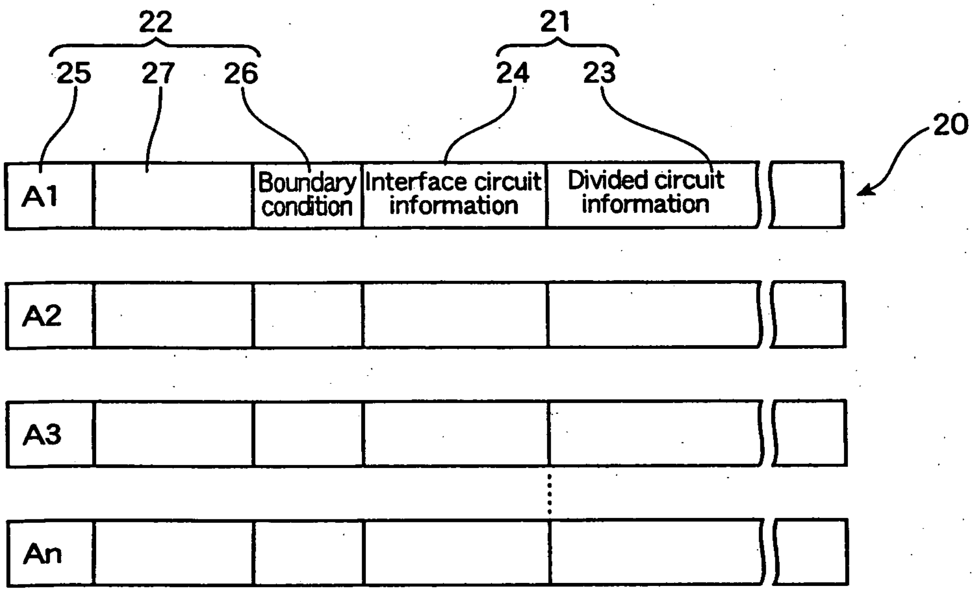

22
21
25
27
26
24
23
20
A1
Boundary condition
Interface circuit information
Divided circuit information
A2
A3
An

Fig. 6

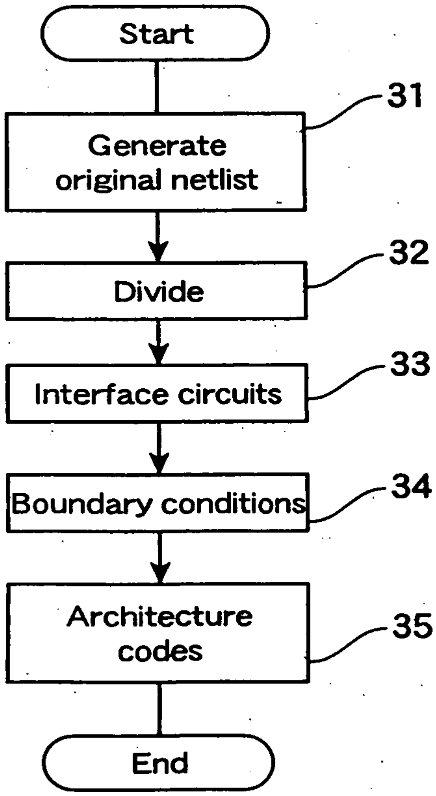

Start
Generate original netlist
31
Divide
32
Interface circuits
33
Boundary conditions
34
Architecture codes
35
End

Fig. 4

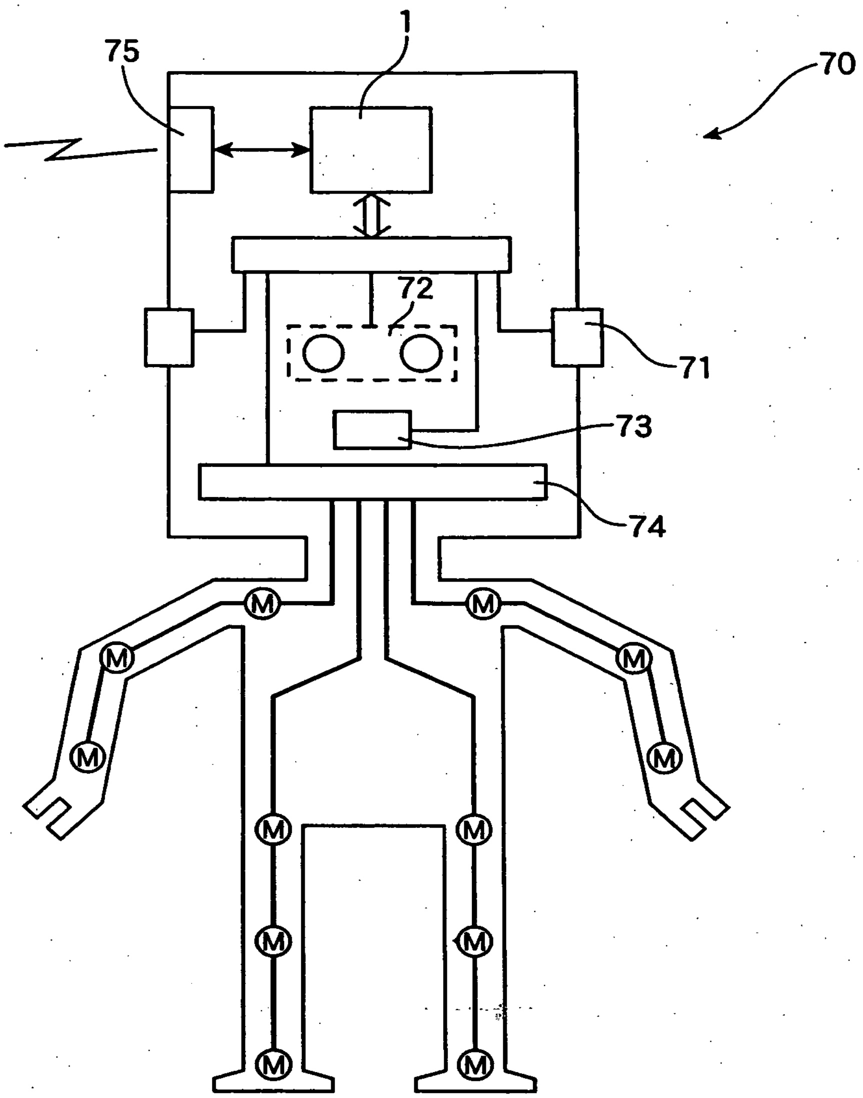


Fig. 5

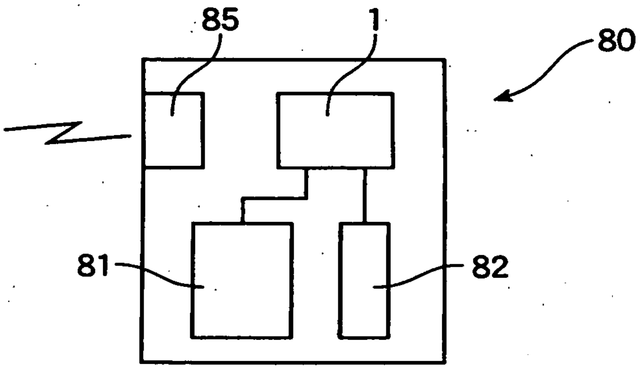


Fig. 7

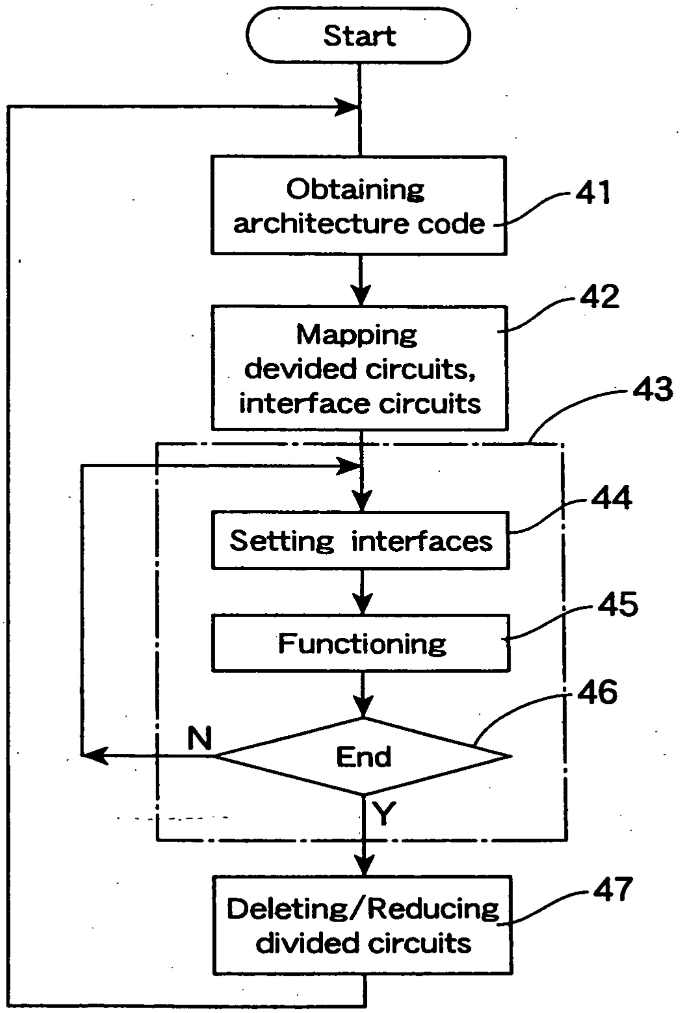

Start
Obtaining architecture code
41
Mapping devided circuits, interface circuits
42
43
Setting interfaces
44
Functioning
45
46
N
End
Y
Deleting/Reducing divided circuits
47

Fig. 8

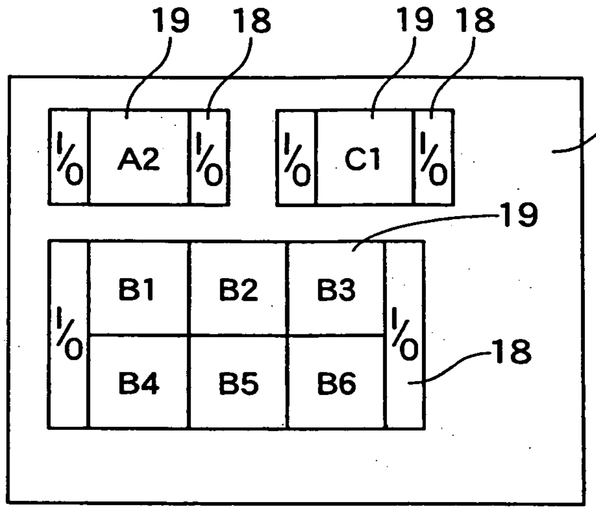
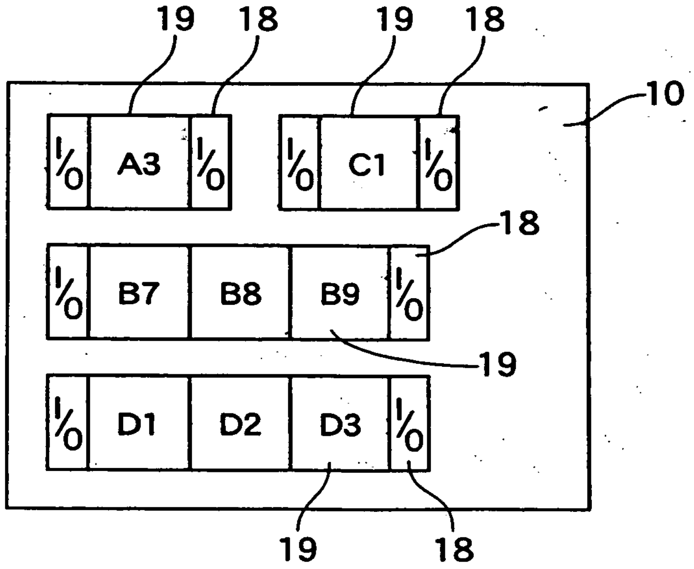

19
18
19
18
10
I/O
A2
I/O
I/O
C1
I/O
19
B1
B2
B3
I/O
I/O
B4
B5
B6
18

Fig. 9


19
18
19
18
10
I/O
A3
I/O
I/O
C1
I/O
18
I/O
B7
B8
B9
I/O
19
I/O
D1
D2
D3
I/O
19
18

Inventor: Tomoyoshi Sato
Title: "Data Processing System"
U.S. Serial No.: Not Yet Assigned
Atty. Docket No.. 29898/40913
Marshall, Gerstein & Borun LLP 312-474-6300
Sheet 7 of 12

Fig. 10

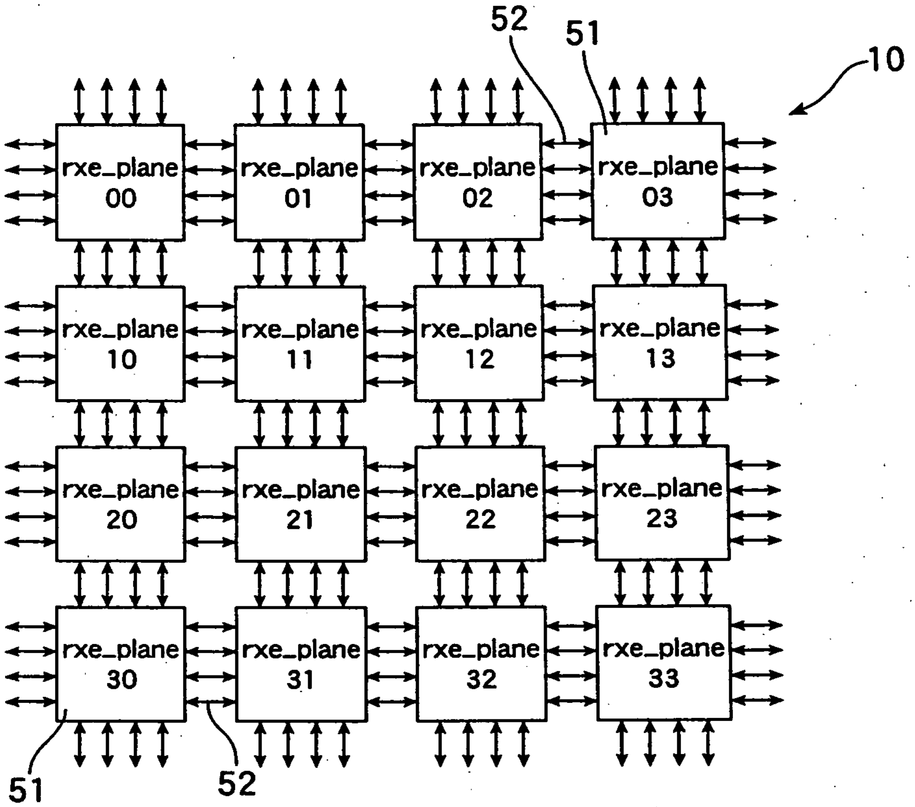
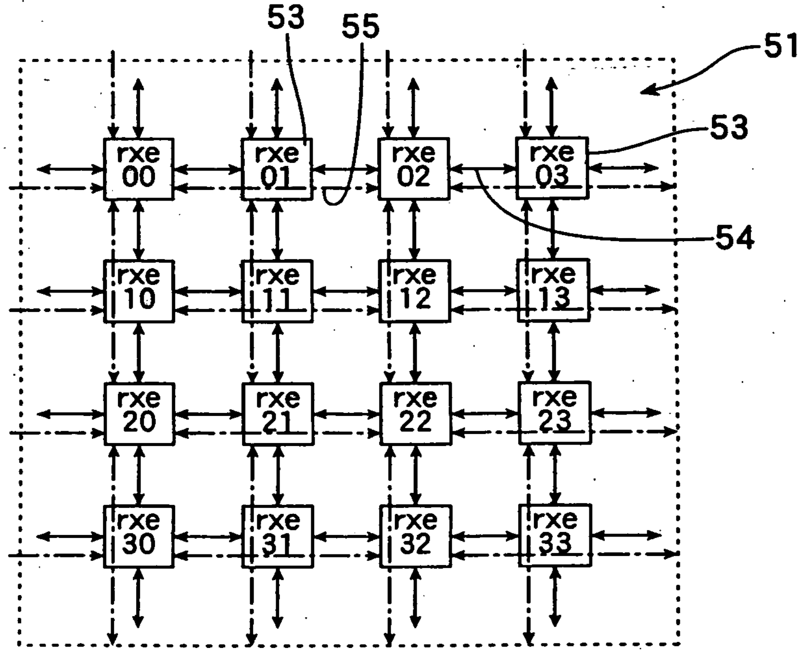


Fig. 11



Inventor: Tomoyoshi Sato
Title: "Data Processing System"
U.S. Serial No.: Not Yet Assigned
Atty. Docket No.. 29898/40913
Marshall, Gerstein & Borun LLP 312-474-6300
Sheet 8 of 12

Fig. 12

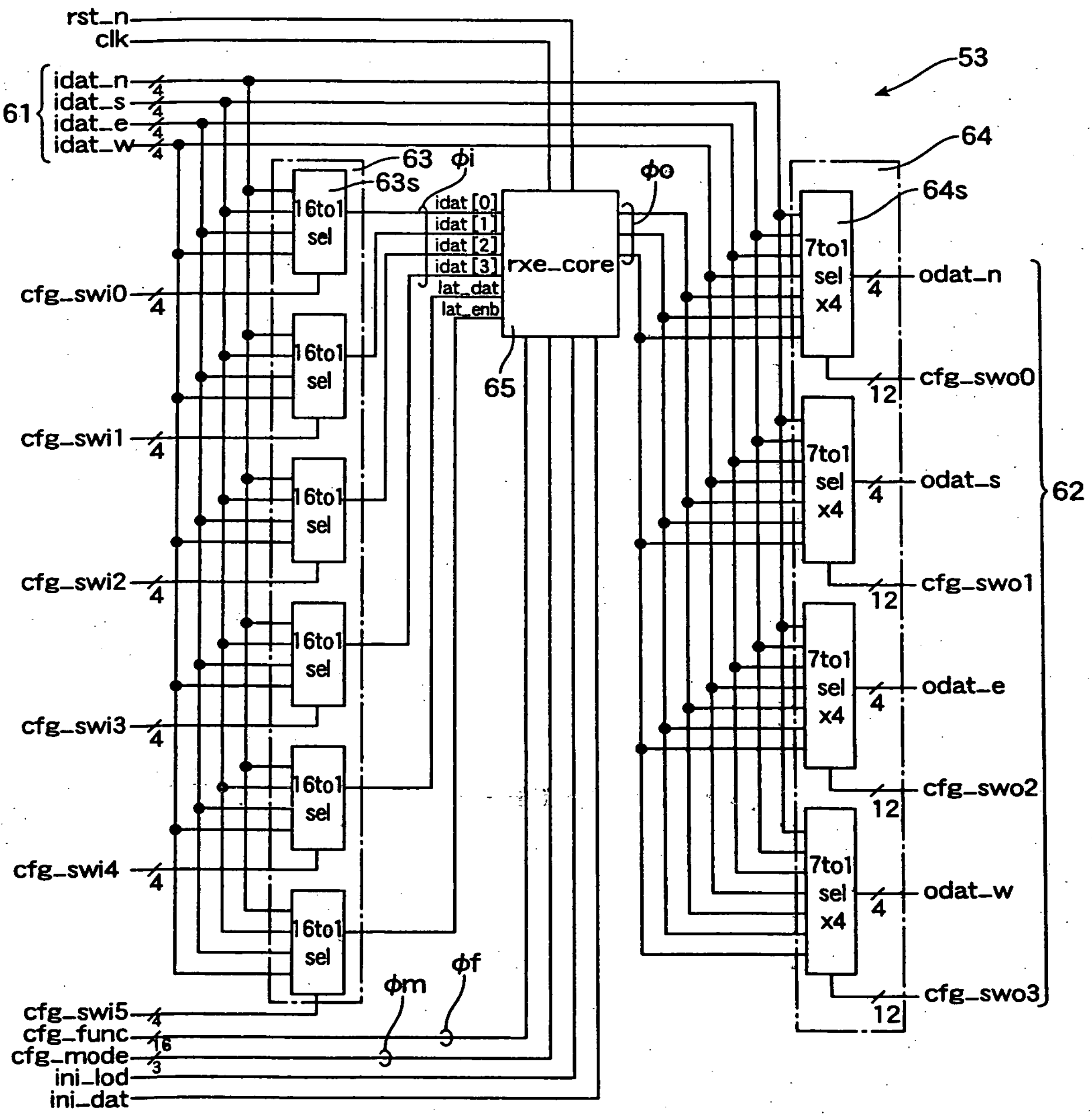


Fig. 13

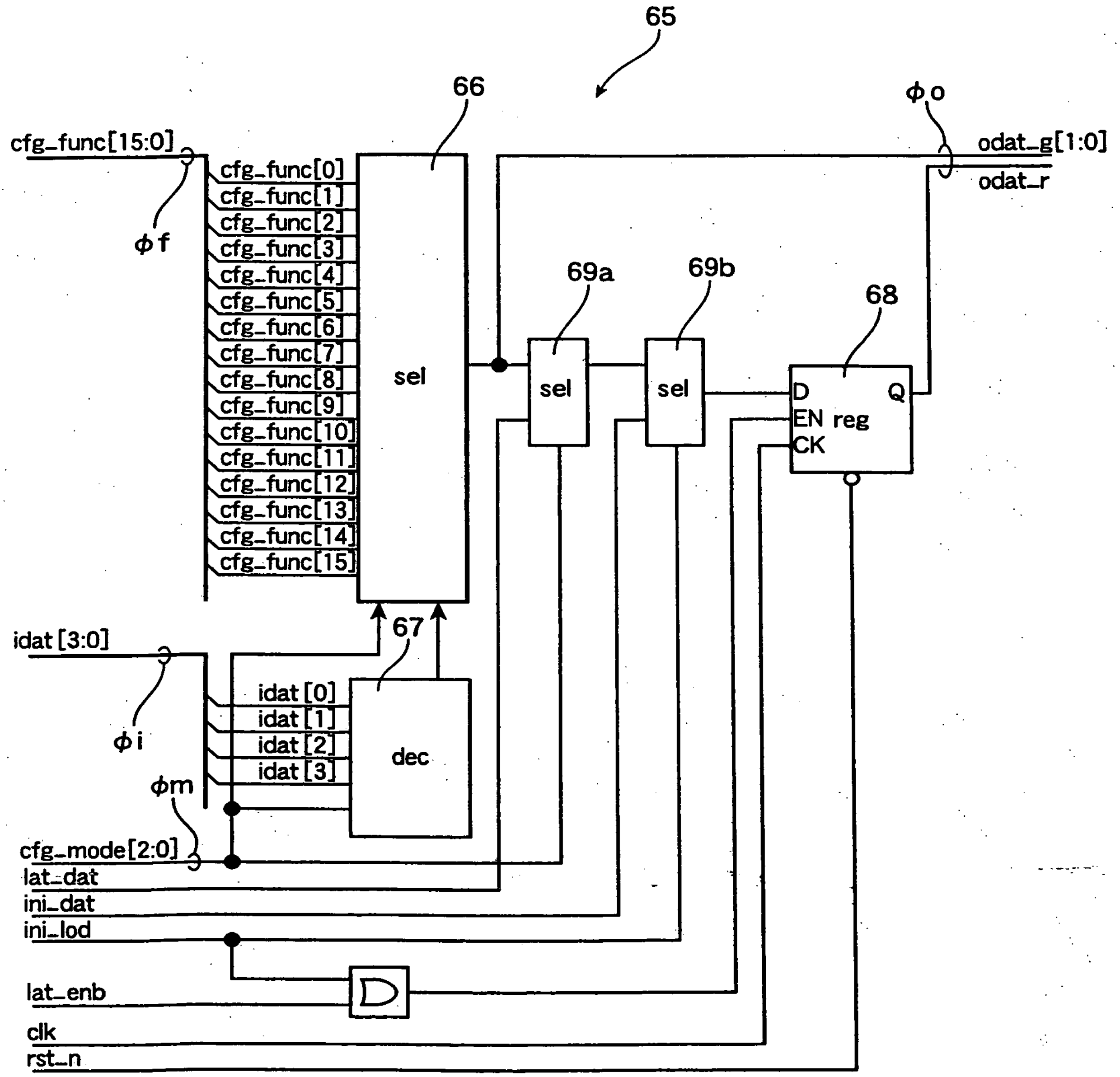

65
66
φo
cfg_func[15:0]
odat_g[1:0]
odat_r
cfg_func[0]
cfg_func[1]
cfg_func[2]
cfg_func[3]
cfg_func[4]
cfg_func[5]
cfg_func[6]
cfg_func[7]
cfg_func[8]
cfg_func[9]
cfg_func[10]
cfg_func[11]
cfg_func[12]
cfg_func[13]
cfg_func[14]
cfg_func[15]
φf
sel
69a
69b
68
sel
sel
D
Q
EN reg
CK
idat [3:0]
67
idat [0]
idat [1]
idat [2]
idat [3]
dec
φi
φm
cfg_mode[2:0]
lat_dat
ini_dat
ini_lod
lat_enb
clk
rst_n

Fig. 14

$\phi$m $\phi$i $\phi$o

| cfg-mod | idat | | | | odat_g | | odat_r | comments |
|---|---|---|---|---|---|---|---|---|
| [2:0] | [3] | [2] | [1] | [0] | [1] | [0] | | |
| 000<br>(4in1out) | 0 | 0 | 0 | 0 | 0 | cfg-func [0] | cfg-func [0] | Hold value of odat-g[0] |
| | 0 | 0 | 0 | 1 | 0 | cfg_func [1] | cfg-func [1] | |
| | 0 | 0 | 1 | 0 | 0 | cfg_func [2] | cfg-func [2] | |
| | 0 | 0 | 1 | 1 | 0 | cfg_func [3] | cfg-func [3] | |
| | 0 | 1 | 0 | 0 | 0 | cfg_func [4] | cfg-func [4] | |
| | 0 | 1 | 0 | 1 | 0 | cfg_func [5] | cfg-func [5] | |
| [MODE 0] | 0 | 1 | 1 | 0 | 0 | cfg_func [6] | cfg-func [6] | |
| | 0 | 1 | 1 | 1 | 0 | cfg_func [7] | cfg_func [7] | |
| | 1 | 0 | 0 | 0 | 0 | cfg_func [8] | cfg_func [8] | |
| | 1 | 0 | 0 | 1 | 0 | cfg_func [9] | cfg_func [9] | |
| | 1 | 0 | 1 | 0 | 0 | cfg_func [10] | cfg_func [10] | |
| | 1 | 0 | 1 | 1 | 0 | cfg_func [11] | cfg_func [11] | |
| | 1 | 1 | 0 | 0 | 0 | cfg_func [12] | cfg_func [12] | |
| | 1 | 1 | 0 | 1 | 0 | cfg_func [13] | cfg_func [13] | |
| | 1 | 1 | 1 | 0 | 0 | cfg_func [14] | cfg_func [14] | |
| | 1 | 1 | 1 | 1 | 0 | cfg_func [15] | cfg_func [15] | |
| 001<br>(4in1out) | 0 | 0 | 0 | 0 | 0 | cfg_func [0] | lat_dat | Split register |
| | 0 | 0 | 0 | 1 | 0 | cfg_func [1] | lat_dat | |
| | 0 | 0 | 1 | 0 | 0 | cfg_func [2] | lat_dat | |
| | 0 | 0 | 1 | 1 | 0 | cfg_func [3] | lat_dat | |
| | 0 | 1 | 0 | 0 | 0 | cfg_func [4] | lat_dat | |
| | 0 | 1 | 0 | 1 | 0 | cfg_func [5] | lat_dat | |
| | 0 | 1 | 1 | 0 | 0 | cfg_func [6] | lat_dat | |
| [MODE 1] | 0 | 1 | 1 | 1 | 0 | cfg_func [7] | lat_dat | |
| | 1 | 0 | 0 | 0 | 0 | cfg_func [8] | lat_dat | |
| | 1 | 0 | 0 | 1 | 0 | cfg_func [9] | lat_dat | |
| | 1 | 0 | 1 | 0 | 0 | cfg_func [10] | lat_dat | |
| | 1 | 0 | 1 | 1 | 0 | cfg_func [11] | lat_dat | |
| | 1 | 1 | 0 | 0 | 0 | cfg_func [12] | lat_dat | |
| | 1 | 1 | 0 | 1 | 0 | cfg_func [13] | lat_dat | |
| | 1 | 1 | 1 | 0 | 0 | cfg_func [14] | lat_dat | |
| | 1 | 1 | 1 | 1 | 0 | cfg_func [15] | lat_dat | |

Fig. 15

| cfg-mod | idat | | | | odat_g | | odat-r | comments |
|---|---|---|---|---|---|---|---|---|
| [2:0] | [3] | [2] | [1] | [0] | [1] | [0] | | |
| 010<br>(2in1out)<br>[MODE 2] | | | 0 | 0 | | cfg_func [0] | cfg_func [0] | Highest two bits and lowest 2 bits form two separate series. Register holds result for lowest 2 bits. |
| | | | 0 | 1 | | cfg_func [1] | cfg_func [1] | |
| | | | 1 | 0 | | cfg_func [2] | cfg_func [2] | |
| | | | 1 | 1 | | cfg_func [3] | cfg_func [3] | |
| | 0 | 0 | | | cfg_func [8] | | | |
| | 0 | 1 | | | cfg_func [9] | | | |
| | 1 | 0 | | | cfg_func [10] | | | |
| | 1 | 1 | | | cfg_func [11] | | | |
| 011<br>(2in1out)<br>[MODE 3] | | | 0 | 0 | | cfg_func [0] | | Highest two bits and lowest 2 bits form two separate series. Register holds result for highest 2 bits. |
| | | | 0 | 1 | | cfg_func [1] | | |
| | | | 1 | 0 | | cfg_func [2] | | |
| | | | 1 | 1 | | cfg_func [3] | | |
| | 0 | 0 | | | cfg_func [8] | | cfg_func [8] | |
| | 0 | 1 | | | cfg_func [9] | | cfg_func [9] | |
| | 1 | 0 | | | cfg_func [10] | | cfg_func [10] | |
| | 1 | 1 | | | cfg_func [11] | | cfg_func [11] | |
| 100<br>(2in1out)<br>[MODE 4] | | | 0 | 0 | | cfg_func [0] | lat_dat | Highest two bits and lowest 2 bits form two separate series. Split register. |
| | | | 0 | 1 | | cfg_func [1] | lat_dat | |
| | | | 1 | 0 | | cfg_func [2] | lat_dat | |
| | | | 1 | 1 | | cfg_func [3] | lat_dat | |
| | 0 | 0 | | | cfg_func [8] | | lat_dat | |
| | 0 | 1 | | | cfg_func [9] | | lat_dat | |
| | 1 | 0 | | | cfg_func [10] | | lat_dat | |
| | 1 | 1 | | | cfg_func [11] | | lat_dat | |
| 101<br>(3in1out)<br>[MODE 5] | x | 0 | 0 | 0 | cfg_func [8] | cfg_func [0] | cfg_func [0] | MSB unused. Hold value of odat-g[0] |
| | | 0 | 0 | 1 | cfg_func [9] | cfg_func [1] | cfg_func [1] | |
| | | 0 | 1 | 0 | cfg_func [10] | cfg_func [2] | cfg_func [2] | |
| | | 0 | 1 | 1 | cfg_func [11] | cfg_func [3] | cfg_func [3] | |
| | | 1 | 0 | 0 | cfg_func [12] | cfg_func [4] | cfg_func [4] | |
| | | 1 | 0 | 1 | cfg_func [13] | cfg_func [5] | cfg_func [5] | |
| | | 1 | 1 | 0 | cfg_func [14] | cfg_func [6] | cfg_func [6] | |
| | | 1 | 1 | 1 | cfg_func [15] | cfg_func [7] | cfg_func [7] | |
| 110<br>(3in1out)<br>[MODE 6] | x | 0 | 0 | 0 | cfg_func [8] | cfg=func [0] | cfg_func [8] | MSB unused. Hold value of odat-g[1] |
| | | 0 | 0 | 1 | cfg_func [9] | cfg_func [1] | cfg_func [9] | |
| | | 0 | 1 | 0 | cfg_func [10] | cfg_func [2] | cfg_func [10] | |
| | | 0 | 1 | 1 | cfg_func [11] | cfg_func [3] | cfg_func [11] | |
| | | 1 | 0 | 0 | cfg_func [12] | cfg_func [4] | cfg_func [12] | |
| | | 1 | 0 | 1 | cfg_func [13] | cfg_func [5] | cfg_func [13] | |
| | | 1 | 1 | 0 | cfg_func [14] | cfg_func [6] | cfg_func [14] | |
| | | 1 | 1 | 1 | cfg_func [15] | cfg_func [7] | cfg_func [15] | |
| 111<br>(3in1out)<br>[MODE 7] | x | 0 | 0 | 0 | cfg_func [8] | cfg_func [0] | lat_dat | MSB unused. Split register |
| | | 0 | 0 | 1 | cfg_func [9] | cfg_func [1] | lat_dat | |
| | | 0 | 1 | 0 | cfg_func [10] | cfg_func [2] | lat_dat | |
| | | 0 | 1 | 1 | cfg_func [11] | cfg_func [3] | lat_dat | |
| | | 1 | 0 | 0 | cfg_func [12] | cfg_func [4] | lat_dat | |
| | | 1 | 0 | 1 | cfg_func [13] | cfg_func [5] | lat_dat | |
| | | 1 | 1 | 0 | cfg_func [14] | cfg_func [6] | lat_dat | |
| | | 1 | 1 | 1 | cfg_func [15] | cfg_func [7] | lat_dat | |

Fig. 16

| Function | cfg-mode [2:0] | cfg-func [15:0] | comments |
|---|---|---|---|
| Inverter | 010/011/100 | xxxx_xxxx_xxxx_0101 | Use lowest bit |
| 2-input AND | 010/011/100 | xxxx_xxxx_xxxx_1000 | Use lowest 2 bits |
| 2-input NAND | 010/011/100 | xxxx_xxxx_xxxx_0111 | Use lowest 2 bits |
| 2-input OR | 010/011/100 | xxxx_xxxx_xxxx_1110 | Use lowest 2 bits |
| 2-input NOR | 010/011/100 | xxxx_xxxx_xxxx_0001 | Use lowest 2 bits |
| 2-input EXOR | 010/011/100 | xxxx_xxxx_xxxx_0110 | Use lowest 2 bits |
| 2-input EXNOR | 010/011/100 | xxxx_xxxx_xxxx_1001 | Use lowest 2 bits |
| 3-input AND | 101/110/111 | xxxx_xxxx_1000_0000 | Use lowest 3 bits |
| 3-input NAND | 101/110/111 | xxxx_xxxx_0111_1111 | Use lowest 3 bits |
| 3-input OR | 101/110/111 | xxxx_xxxx_1111_1110 | Use lowest 3 bits |
| 3-input NOR | 101/110/111 | xxxx_xxxx_0000_0001 | Use lowest 3 bits |
| FullAdder | 101/110/111 | 1110_1000_1001_0110 | Use lowest 3 bits. Carry in highest output bits. Sum in lowest output bits. |
| 4-input AND | 000/001 | 1000_0000_0000_0000 | |
| 4-input NAND | 000/001 | 0111_1111_1111_1111 | |
| 4-input OR | 000/001 | 1111_1111_1111_1110 | |
| 4-input NOR | 000/001 | 0000_0000_0000_0001 | |
| 4-input EXOR | 000/001 | 0111_1111_1111_1110 | |
| 4-input NOR | 000/001 | 1000_0000_0000_0001 | |
| AND_AND_OR | 000/001 | 1000_1000_1000_1000 | |
| AND_AND_NOR | 000/001 | 0111_0111_0111_0111 | |
| 4-input comparator(1111) | 000/001 | 1000_0000_0000_0000 | Values to be compared set at 1 |